International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN (P): 2249-6831; ISSN (E): 2249-7943
Vol. 8, Issue 2, Jun 2018, 83-90

EVOLUTIONARY COMPUTATION BASED OPTIMIZATION TECHNIQUE FOR MINIMUM VERTEX COVER

SHET RESHMA PRAKASH

CMR Institute of Technology, Department of Computer Science and Engineering, Bangalore, Karnataka, India

ABSTRACT

Minimum Vertex Cover (MVC) is a classical problem of graph theory and belongs to the class of NP-Complete, hence finding the approximate solution is much more efficient compared to the exact solution. In this paper, the proposed solution is designed to find the approximate solution of Minimum Vertex Cover problem for the un-weighted graph. The solution is designed and implemented using evolutionary computation based Genetic Algorithm (GA). The performance of proposed solution is tested on various DIMACS benchmark graph instances. The experimental results obtained are very accurate for the graph instances having less than or equal to 500 vertices.

KEYWORDS*: *Evolutionary Computation, Minimum Vertex Cover, Genetic Algorithm, Dimacs & Computation

Received: Apr 16, 2018**; Accepted:** May 07, 2018**; Published:** May 23, 2018**; Paper Id.:** IJCSEITRJUN201810

Original Article

INTRODUCTION

NP-Hard problems are those in which you cannot compute the solution in polynomial time. Also finding the exact solution in polynomial time is difficult for NP-Complete problems. Hence it is better to use the approximation algorithm which finds the approximate solution close to an optimal solution for such NP-Complete problems. Vertex Cover problem belongs to the field of graph theory and it is a collection of a set of vertices which covers all edges of the graph. There are two types of Vertex Cover problem i.e. decision-based and optimization based. The decision-based problem is to determine if there exists a vertex cover of the specified size whereas the minimum vertex cover is an NP-Complete optimization problem in the field of computer science. The minimum vertex cover for an unweighted graph is the smallest group of distinct vertices that cover all the edges of the graph. Consider a graph G= {V, E} where V is set of vertices and E is set of edges. Hence the vertex cover for graph G is subset **S i.e. S ⊆ V if ∀{a, b} ∈ E: a ∈ S ∨ b ∈ S.**

Hence the minimum vertex cover [1] for Graph G is vertex cover S that minimizes |S|. Consider Figure 1 given below where two possible vertex cover is shown in red color. But the minimum vertex cover for this is in the one with two vertices.

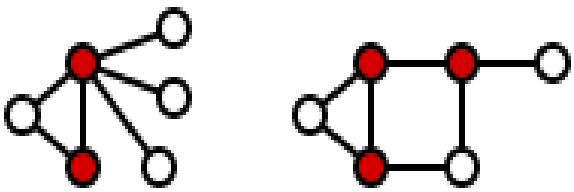

Figure 1: Vertex Cover for Sample Graph

In this paper, the proposed solution is designed and implemented using Genetic Algorithm [9]. Genetic algorithm belongs to the area of evolutionary algorithms and is used to find an optimal solution through the process of evolution. There are many existing approximation algorithms developed so far for solving the minimum vertex cover. Jingrong Chen et al. [1] have proposed an approximation algorithm based on Dijkstra's algorithm for finding minimum vertex cover. In this paper [1] the different vertex cover is obtained by considering various initial points and the time complexity is O (n^3). Mohammed Eshtay et al. [2] have proposed a greedy solution called NMVSA (New Modified Vertex Support Algorithm) for finding the minimum vertex cover of the graph. In this paper [2] they have compared NMVSA with other algorithms like VSA [3], MVSA [4] and tested the effectiveness of algorithm on 40 instances of BHOSLIB and 13 instances of DIMACS [5] dataset. S. Balaji et al. [3] proposed a new algorithm called VSA (Vertex Support algorithm) and tested its performance by considering random graphs of different sizes and DIMACS benchmark graphs. The paper is organized as follows. Section 1 gives the basic introduction of the paper. Section 2 and 3 describes the conceptual details of Evolutionary Computation and Genetic Algorithm. Section 3 outlines the proposed solution. Section 4 describes the proposed algorithm. Section 5 outlines the steps required for the proposed algorithm. Section 6 lists the simulation results obtained by testing the proposed solution with respect to different DIMACS benchmark graph instances. Section 7 concludes the paper.

EVOLUTIONARY COMPUTATION

Evolutionary computation [10] belongs to the field of computer science and enables optimization of real-world problems by following the principles of biological evolution. The first step is to define the objective function which is either minimized or maximized according to the problem domain. Next, it involves generating candidate solutions from initial population and also continuously improving and updating it across many generations. In each generation, the less desired solutions are removed and some random changes are made to selected candidate solutions to get a final optimized solution. The field of Evolutionary Computation involves many algorithms like Genetic Algorithm (GA), Differential Evolution (DE), Genetic Programming, Evolutionary Programming, Gene Programming and many more.

GENETIC ALGORITHM

A genetic algorithm was first introduced by John Holland and is inspired by Darwinian's theory of survival of fittest. It supports the evolutionary computation through a combination of three steps namely parent selection, crossover and mutation. The whole process of evolution starts by searching the initial population and randomly generating set of potential solutions called as chromosomes. The fitness value is calculated for each chromosome by using the fitness function defined for the real world problem. The chromosomes with higher fitness value are selected as parents who will produce offspring. Also, some modifications are made to the offspring and it is considered for further optimization. The whole evolution process is carried out repeatedly until the termination criteria defined for objective function is satisfied. It includes following ideas.

Initialization

According to the problem domain, the population size [7] will be initialized. Also, initial population or chromosomes are generated in a random fashion by exploring the whole search space and by using pseudo-random numbers. Each chromosome contains the fixed number of genes and the value of genes can be represented in binary, real or integer form.

Fitness Calculation

The fitness value is calculated for each chromosome by processing the gene structure with respect to some pre-defined fitness function.

Selection, Mutation and Crossover

The selection [9] procedure will select two chromosomes with highest fitness value. There are various methods for selecting chromosomes like Fitness Proportionate selection (i.e. Roulette Wheel Selection), Boltzmann's Selection, Tournament Selection and many more. In case of Crossover, the parent chromosomes would reproduce the offspring chromosome. There are many methods for performing crossovers [9] like uniform crossover, one point crossover, multi-point crossover, arithmetic crossover and many more. In case of mutation [9] the genes contained in offspring chromosomes (i.e. intermediate solutions) are altered to increase its fitness value. The intermediate solutions are further considered for the whole optimization process. Hence the cycle of selection, mutation and crossover is repeated for many generations until it meets the termination criteria. The termination criteria should be decided carefully for proper convergence of the final optimized solution.

PROPOSED SOLUTION

In this paper, a novel technique is proposed to generate the Minimum Vertex Count for very large graph instances. The proposed solution uses Genetic Algorithm to find the approximate solution for Minimum Vertex Cover problem which has many applications in real life. One possible application is to determine the optimal solution for placement of sensors, ATM's and other facilities across different cities. An initial population consists of some random Graph G = {V, E} where V is a count of vertices and E denotes the count of edges. The population size is assumed as V/10. Hence if there are 400 vertices the population size will be 40. The number of chromosomes generated in each generation is equal to population size. The chromosomes are encoded by value encoding and it contains the string of integer values which represents the vertices of the graph. Value encoding is useful because each value is important for finding the fitness of the chromosome. The chromosomes are generated randomly and fitness is evaluated using following fitness function.

$$\text{Fitness } (c_i) = \sum_{i=1}^{population\_size} \text{Degree } (ci)/\text{Total} \quad (1)$$

$$\text{Degree } (c_i) = \sum_{i=1}^{V/population\_size} degree(g_i) \quad (2)$$

$$\text{Total} = \sum_{i=1}^{population\_size} \text{Degree } (c_i) \quad (3)$$

In the above equation, c_i denotes the chromosome and g_i represent each gene of the chromosome. The chromosomes with highest fitness value are selected for reproduction using Roulette Wheel Selection [9] method. The Roulette Wheel Selection procedure is a stochastic algorithm which is based on the concept of survival of fittest. The whole process starts by first finding the selection probability and cumulative sum for each individual chromosome according to its fitness value. The Table 1 given below shows the selection probability and fitness for each individual chromosome. Selection probability is calculated using following formula.

$$\text{Selection probability } (c_i) = \frac{\text{Fitness } (ci)}{\sum_{i=1}^{population\_size} Fitness(ci)} \quad (4)$$

Table 1: Sample Data for Five Chromosomes with Fitness Value

Chromosome	Fitness	Selection Probability	Cumulative Sum
1	2.5	0.21	0.21
2	3.5	0.29	0.50
3	4.5	0.38	0.88
4	1.0	0.08	0.96
5	0.5	0.04	1.0

Figure 2: Roulette Wheel for Sample Data

As shown in table 1. the chromosome 2 and 3 has highest fitness value and hence it has a high chance of getting selected for mating since it occupies the largest share in the Roulette Wheel. The Roulette Wheel for sample data of table 1 is shown in Figure 2 given below. Consider the following line segment shown in Figure 3 where all the chromosomes are mapped and each chromosome's segment area is equal to its fitness value.

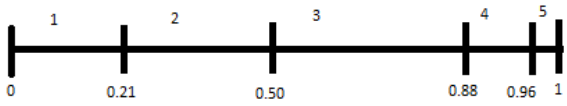

Figure 3: Line Segment Showing Fitness Value

Next step is to generate a random number between 0.0 and 1.0 since the cumulative sum is under the range of 1.0. Thus if the random number generated is 0.51 then chromosome 3 is selected for the mating. The chromosome with higher fitness value has large shares in the roulette wheel and hence they have more chances of survival for the next generation. The selected chromosomes reproduce using a uniform crossover to generate two offspring chromosome. Both the offspring chromosomes are formed by selecting fifty percent of the genes from each parent chromosome. The offspring chromosome is further subjected to mutation. Mutation is performed by random resetting where a gene is altered randomly with another gene in order to increase the final fitness value of the offspring. The offspring chromosome with the highest fitness is selected as an intermediate solution and the whole procedure of selection, crossover and mutations is repeated for V/10 generations.

PROPOSED ALGORITHM

The algorithm for the proposed solution is given below

Step 1: Input Graph G= {V, E} where V is set of vertices and E is a number of edges. Set initial value for mutation probability and population size.

Step 2: Calculate and update the degree of each vertex.

Step 3: Randomly generate m chromosomes (i.e. candidate solutions) where m=1 to population size. Calculate fitness for each chromosome.

Step 4: Use Roulette Wheel Selection (RWS) method to select two chromosomes as parents.

Step 5: Generate offspring by uniform crossover method and also perform the mutation for each offspring. Random resetting strategy is used to perform mutation.

Step 6: The offspring with the highest fitness is selected and is further considered for the whole optimization process. Update Minimum Vertex Count List.

Step 7: For n generations repeat Steps 2, 3, 4, 5 and 6 where n varies from 1 to population size. After every 10 generations, the mutation probability is reduced by 0.1.

Step 8: Add each remaining vertex to Minimum Vertex Count List and repeat step 2. Obtain final optimized solution.

EXPERIMENTAL RESULTS

Table 2: Simulation Results for DIMACS Graph Instances

Benchmark	No of Vertices	Optimal V_c	Observed V_c
brock200_1	200	179	179
brock200_2	200	188	187
brock200_3	200	185	185
brock200_4	200	183	182
brock400_3	400	369	369
c-fat200-1	200	188	183
c-fat200-2	200	176	176
c-fat500-5	500	446	446
gen400_p0.9_55	400	345	345
gen400_p0.9_65	400	335	335
hamming6-2	64	60	57
hamming8-4	256	240	232
Johnson8-2-4	28	24	24
Johnson16-2-4	120	112	112
Keller4	171	160	158
p-hat300-2	300	275	267
p-hat300-3	300	274	274
san200-0.7.2	200	182	181
san400-0.7.1	400	360	355

Table 3: Approximation Ratio for DIMACS Graph Instances

Benchmark	Approximation Ratio
brock200_1	1
brock200_2	1.005347594
brock200_3	1
brock200_4	1.005494505
brock400_3	1
c-fat200-1	1.027322404
c-fat200-2	1
c-fat500-5	1
gen400_p0.9_55	1

Table 3: Contd.,	
gen400_p0.9_65	1
hamming6-2	1.052631579
hamming8-4	1.034482759
Johnson8-2-4	1
Johnson16-2-4	1
Keller4	1.012658228
p-hat300-2	1.029962547
p-hat300-3	1
san200-0.7.2	1.005524862
san400-0.7.1	1.014084507

The performance of proposed solution is tested on various DIMACS benchmark graph instances taken from DIMACS implementation challenge suit. The results obtained are shown in Table 2. The first and second column lists the name and count of vertices for benchmark graph instances. The third column represents the known optimal V_c obtained from the DIMACS implementation challenge. The fourth column represents the Observed results of the proposed solution. The performance is tested by comparing the Optimal V_c and Observed V_c. The accuracy of the result is shown in Table 3. The accuracy is measured by calculating the approximation ratio. The approximation ratio is obtained by taking the ratio of the Observed V_c and Optimal V_c. It is observed that the approximation ratio does not exceed the value 1.05.

CONCLUSIONS

The proposed solution is implemented using Java Programming Language on Net Beans IDE version 8.2. In conducting this experiment the parameters were set as follows. The population size is initialized as V/10 in order to maintain it in the range of 20 to 50. Also, high mutation probability of 0.7 is set at the beginning which is further reduced by 0.1 after every 10 generations. The proposed solution efficiency is measured by considering different benchmark graph instances of different cardinality and known optimal V_c. By analyzing the experimental results it can be concluded that the proposed solution gives accurate results for the different graph instances of varied sizes. In future, efforts will be made to extend this problem to solve the minimum vertex cover for the weighted graph.

REFERENCES

1. *Jingrong Chen, Lei Kou, Xiaochuan Cui, An Approximation Algorithm for the Minimum Vertex Cover Problem, Procedia Engineering 137 (2016) 180 – 185*
2. *Mohammed Eshtay, Azzam Sleit, Ahmad Sharieh, NMVSA Greedy Solution for Vertex Cover Problem in International Journal of Advanced Computer Science and Applications, Vol. 7, No. 3, 2016*
3. *S. Balaji, V. Swaminathan and K. Kannan, Optimization of Un-weighted Minimum Vertex Cover, International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering Vol:4, No:7, 2010*
4. *Khan Imran1 and Khan Hasham, Modified Vertex Support Algorithm: A New approach for approximation of Minimum vertex cover, Research Journal of Computer and Information Technology Sciences, Vol.* ***1(6),*** *7-11, November* ***(2013).***
5. *DIMACS clique benchmarks, Benchmark instances made available by electronic transfer at dimacs.rutgers.edu, Rutgers Univ., Piscataway. NJ. (1993).*
6. *Majhi, Bidya Prakash, and Shatendra Sahu. "A rewiew on Application of Bio-Geography based Algorithm and Other Optimization Techniques."*

7. *Nitasha Soni, Dr 2 Tapas Kumar, Study of Various Mutation Operators in Genetic Algorithms,) International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, 4519-4521.*
8. *Olympia Roeva, Stefka Fidanova, Marcin Paprzycki, Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling, Proceedings of the 2013 Federated Conference on Computer Science and Information Systems pp. 371–376.*
9. *Moosazadeh, Sayfoddin, and Farshad N. Shahmohamad. "Drilling Optimization using Minimum Energy Concept."*
10. *S. J. Shyu, P. Y. Yin and B. M. T. Lin, An ant colony optimization algorithm for the minimum weight vertex cover problem, Annals of Operations Research, Vol. 131, (2004), 283 - 304.*
11. *Rahul Malhotra, Narinder Singh & Yaduvir Singh, Genetic Algorithms: Concepts, Design for Optimization of Process Controllers, Published by Canadian Center of Science and Education 39, Vol. 4, No. 2; March 2011.*
12. *A.E. Eiben, M. Schoenauer, Evolutionary computing, Information Processing Letters 82 (2002) 1–6, Elsevier.*